教育諸学研究　第36巻　44－64　2023

研究論文

小学校教員に求められるプログラミングスキルとその利活用
―自らのプログラミングスキル習得過程の分析からの考察―

坪　谷　諭泉那[1]・穴　田　恭　輔[2]

はじめに

2020年度から、小学校ではプログラミング教育が実施されている。それに先立ち，2016年12月の中央教育審議会答申（平成28年答申）[1]で、学校において育成を目指す資質・能力として、言語能力と情報活用能力が例示的に示された。その一つめの言語能力については、明らかに言葉は学習活動を支える上で重要な役割を果たしている。人間がものごとを考えたり、それを表現したりするときには「言葉による思考」が働いている。私たちが認識した情報を基に思考し、それを表現していく過程では常に言葉が媒介している。言語能力は全ての教科等の学習の基盤であり、国語と外国語だけでなく全ての教科等の教育を通して、より一層の充実を図ることが必要である。二つめの情報活用能力については、現代は情報化が急速に進展し、身の回りのものに情報技術が活用されていることから、生活上必要な手続き、日常生活における営みは情報技術を通じて行うことが当たり前で、そのような時代にふさわしい情報モラルも含めた情報技術の活用能力が求められている。身近なものにコンピュータが内蔵され、それらがプログラムの働きにより制御されていることを子供が知り、小・中・高等学校を通じたプログラミング教育の充実を図ることは現代の課題である。プログラミング教育の実施について、平成28年答申では「小・中・高等学校を見通した学びの過程の中で、『主体的・対話的で深い学び』の実現に資するプログラミング教育とすることが重要である。」とされている。それは、必要があれば、プログラミング教育で培ったその力を主体的に、積極的に、利活用できるようなプログラミング教育でなければならないことを意味すると考えられる。

そこで本研究では、言語能力、情報活用能力に資するプログラミング教育について考える。そしてそのための小学校におけるプログラミング教育の在り方の一つとして、小学校教員にもプログラミングスキルが求められるという仮説を立て、プログラミング未経験者であった本稿筆頭著者を学習者としてプログラミングスキル習得過程を示すとともに、それを習得した立場から、小学校におけるプログラミング教育の指導者にはどの程度のスキルが求められ、何が必要であるかを考察する。

プログラミングは、順次、分岐、反復の3つの制御構造からなる。構造化定理によるとこれらの3つの制御の組み合わせですべてのアルゴリズムが記述できる。小学校のプログラミング教育のねらいにプログラミング的思考[2]の育成があるが、それは「自分が意図する一連の活動を実現するために、どのような動きの組み合わせが必要であり、一つ一つの動きに対応した記号を、どのように組み合わ

[1] 神戸女子大学学校教育学専攻科修了生，淡路市立石屋小学校
[2] 神戸女子大学

せたらいいのか、記号の組み合わせをどのように改善していけば、より意図した活動に近づくのか、といったことを論理的に考えていく力」とされている。それは自分が意図する一連の活動をプログラミングの３つの制御構造である順次、分岐、反復だけで構成していく力であり、その具体化の一つはプログラミング言語による表現である。これも広い意味では、言語による表現である。したがって、言語能力の育成は国語や外国語のような日常言語だけでなく、プログラミングによる思考活動でも育成されるという観点からの考察も行う。

第１章　本研究の目的と方法

第１節　問題提起

小学校でプログラミング教育が必修化され、教員にもプログラミングについての知識やスキルが必要であると考える。それは教員自身にそれらが無いと、児童にプログラミングについて知ってもらいその良さを感じてもらうことができないし、ましてや、プログラミング教育のねらいにあげられているプログラミング的思考の育成や、各教科の学びをより確かなものにするために適切にプログラミングを取り入れることなどは、到底できるはずもないからである。

本稿筆頭著者がそのことを実感したのは、担当している小学５年生の図画工作の授業で、担任の先生から、プログラミングに触れる機会を作ってあげたらどうかとアドバイスを受けて行った授業である。このとき使用したアプリケーションは Benesse の「Codeable Crafts」である。このアプリケーションでは、犬やくじらなどの動物の台紙に自由に色を塗ったり、アイテムで飾り付けをしたりすることで、自分だけのキャラクターを作ることができる。また、完成したオリジナルのキャラクターをプログラムで自由に動かすことで、自分だけのデジタル絵本を作成することができるというものである。

児童らは、前に一度このアプリケーションを使ったことがあるようで、操作には迷いがなく自分だけのキャラクターデザインには取り組むことができていた。しかし、キャラクター完成後に、それらの動きのプログラムを考えたり、実際に動かしたりする児童はいなかった。このようになった原因は、指導者自身が事前にキャラクターの動かし方のパターンのいくつかをサンプルプログラムとして示し、児童がそれらを自由に組み合わせて使えるような準備を怠ったからである。児童が勝手にプログラミングすることを期待しても、当然のことながらそれは無理な話である。Scratch でアニメーションを作成するためのプログラミングスキルがあるにも関わらず、そのスキルを役立てることもしなかったことを反省している。以前から小学校教員にもプログラミングスキルは必要と考えていたが、この苦い経験からより一層、児童にプログラミングを体験させたり、各教科に取り入れて実施したりするために、教員自身にもプログラミングの知識やスキルが必要不可欠であると強く考えるようになった。

第２節　目的

以上より、本研究の目的を

(1) 小学校に導入されたプログラミング教育を円滑に実施するために教師にはプログラミングスキル

が必要であると仮定し、どの程度それが必要か、そしてその習得が可能かを確かめること

(2) 人間の思考活動は言語に媒介されて行われるが、プログラミングを学ぶことでも日常の言語能力の育成に寄与するのではないかという仮説を実証的に考察すること

の二点とする。

第３節　研究方法

前節（1）のプログラミングスキルについては、本稿筆頭著者が自ら被験者として、プログラミングの学習者となる。学習者は学部３年次の 2019 年 12 月に１度、正多角形をかくためのプログラミングを体験しているが、その翌年２月から、プログラミングほぼ未経験の状態からプログラミングの学習を開始する。そして本稿最終著者がそのプログラミング指導を行う。本稿では、学習者、指導者と表現する。

学習者：本稿筆頭著者（本稿執筆時には、K 女子大学小学校教員養成課程卒業、学校教育学専攻科修了）

指導者：本稿最終著者

本研究において、学習者はプログラミング言語として次の２つを学習する。

① 2020 年２月～ 2020 年７月まで(学部の３年～４年次)、初めてのプログラミング言語としてビジュアルプログラミング言語　Scratch

その後、

② 2021 年９月～ 2022 年１月まで（専攻科に在学時)、２つ目のプログラミング言語として、テキストプログラミング言語　Python

実際に以上のプログラミング言語を習得した上で、学習者と指導者によって小学校教員に必要なプログラミングスキルについて考察する。 そして、プログラミングを学ぶことによる日常の言語能力への影響についても考察する。

第２章　Scratch とそのプログラミングスキルの習得

第１節　Scratch について

Scratch とは、2006 年にアメリカのマサチューセッツ工科大学メディアラボで開発されたビジュアルプログラミング言語である。プログラムをテキストで記述するのではなく、あらかじめ用意されたブロックを組み合わせることで簡単にプログラミングを始めることができる。また作ったプログラムの結果がすぐに画面で確認でき、プログラムを修正することも簡単であるため、即時的なフィードバック効果も期待できる。とくに Scratch はプログラミング言語教育用の開発環境でもある。

Scratch の画面にはステージがあり、そこに配置するキャラクターのことを「スプライト」という。スプライトを動かしたり、効果音を出したりすることが容易にできるので、好きなキャラクターを用

いたオリジナルのシューティングゲームを作成しながら、プログラミングの練習を始めた。図 1 は、任意の数を 2 で割り切れるかどうかで偶数か奇数かを判別するプログラムである。ねこのスプライトが調べる数を聞いてくるので、調べる整数を入力すると、その数の偶奇を答えてくれる。

図 1　偶数・奇数判定機

第２節　学習の進め方

〔概要〕

［プログラミング言語］　Scratch 3.0

［期間］　2020 年 2 月 13 日～ 2020 年 7 月 1 日

［使用したテキスト］　中植正剛ほか著『Scratch で学ぶプログラミングとアルゴリズムの基本』[3]

［学習の方法］

学習者は、2019 年 12 月 25 日に『小学校プログラミング教育の手引（第二版）』[4] を入手して、学部の卒業論文のテーマには、算数教育とプログラミングを関係づけることを決め、その準備として、自身がプログラミングスキルを習得することから始めた。

学習の方法は、学習者が Scratch のテキストを自習し、そのテキストの課題となっているプログラムを作成し、指導者に提出して指導を受ける。テキストの構成とそこで学習した項目・内容、自習日（学習日）を表 1 に示す。

表 1　使用した Scratch のテキストの構成と学習した内容

	自習日	テキストの章番号と見出し		項目・内容
第 1 週	2020/ 2/13	序章	Scratch の準備	画面の見方
		第 1 章	Scratch の基本	エントリーポイント 逐次処理【順次】 初期化 オブジェクトとメソッド 乱数
		第 2 章	プログラムの流れ	プログラムの流れ ブロードキャスト
第 2 週	2/20 ～	第 3 章	変数と配列	変数 初期化 リスト（配列） エラー処理
		第 4 章	構造化プログラミング	アルゴリズム 条件分岐【分岐】 条件式 入れ子（ネスト）構造 比較演算・論理演算
第 3 週	2/27 ～			繰り返し【反復】 整数と小数の計算 構造化プログラミング
		第 5 章	関数	関数 引数
第 4 週	3/5			再帰処理
		第 6 章	サーチ（探索）	サーチ（探索） 無限ループ
第 5 週	3/16	（第 1 章～第 3 章の提出課題が指導者から返却される）		
第 6 週	3/22	（第 4 章、第 5 章の提出課題が指導者から返却される）		
第 7 週	4/7 ～	（第 1 章から第 6 章の課題プログラムのデバック）		
第 8 週	5/8 ～	第 6 章		最小値のサーチ
		第 7 章	基本的なソート	バブルソート
第 9 週	5/15			選択ソート
第 10 週	5/22			挿入ソート トレース
	7/1	第 8 章	すすんだソート	クイックソート シェルソート
		第 9 章	クローン	クローン

第３節　Scratch 習得の過程

●第 1 週目初日〔2020/2/13〕

プログラミングの自習初日は、テキストの序章、第 1 章、第 2 章を 4 時間ほど集中して自習し、Scratch の基本操作として、プログラムの作り方、スプライトについての追加・変更・複製などの操作やファイルの保存の仕方を知り、プログラミングとしては、逐次処理（順次）、繰り返し（反復）、乱数、オブジェクトとメソッドを使ってシンプルなゲームを作ることができた。スクリプト同士で送り合うメッセージはブロードキャストによって並列処理されたプログラムに渡されることを確認しながらプログラムの流れを把握することができた。

●第 2 週目〔2020/2/20 ～〕

第 3 章、第 4 章の変数とリスト（配列）を 2 ～ 4 時間をそれぞれ 2 日学習し、条件分岐を使うスロットマシーンゲームを作成した。条件分岐では、比較演算、論理演算を行った。

●第 3 週目〔2020/2/27 ～〕

第 4 章の繰り返し、条件分岐を使って、落ち物ゲームを作成し、第 5 章の関数では引数をもつ関数まで行った。

●第 4 週目〔2020/3/5 ～〕

第 5 章の後半にある再帰処理から第 6 章の線形探索まで 5 時間ほど集中して行った。

●第 5 週目〔2020/3/16〕、第 6 週目〔2020/3/22〕

（指導者からのフィードバック）ここまでに学習者が作成したプログラムに対し、指導者からのフィードバックを受け、デバッグをした。

●第 7 週目〔2020/4/7 ～〕

（バグの修正）　1 章～ 6 章の線形探索までのプログラミング課題についてのデバッグを毎回 1 時間で 4 日行った。

●第 8 週目〔2020/5/8 ～〕

数のリストについて、第 6 章では最小値のサーチ、第 7 章ではバブルソートのアルゴリズムの学習をそれぞれ 2 時間ずつで 2 日行った。

●第 9 週目〔2020/5/15〕

第 7 章の選択ソートのアルゴリズムの学習を 2 時間ほど集中して行った。

●第 10 週目〔2020/5/22〕

第 7 章の挿入ソート、トレース、評価の学習を 3 時間ほど集中して行った。

●〔2020/7/1〕

第 8 章のソート、第 9 章のクローンの学習を 5 時間ほど集中して行った。

ただし、第 4 週目以降は、課題の内容も探索やソートであり、初心者にとってはかなり難しく、課題の提出は十分にはできなかった。自習とするにはハードルが高い内容であった。

第４節　Scratch 習得の成果

学習者の Scratch のプログラミングスキルの主な習得成果は、次のとおりである。

(1)　Scratch が得意とするアニメーションで動く簡単なゲームを作成することができるようになった。

(2)　プログラムとプログラミングの算数教育への応用例を算数の単元ごとに考察することができるようになり、単元に合わせていくつかの算数ゲームプログラムも自作した。

(3)　算数の学習とプログラミングの関係をテーマとした学部卒業論文 [5] を書き上げた。
そこでは、面積の単位換算について、面積が $1m^2$ の正方形の一辺は、$1cm^2$ の正方形の一辺の 100 倍であることから、面積は一辺×一辺の 10000 倍になるということを、児童がプログラムを作成することを通して論理的に思考することができるような活用法を提案することができた。

(4)　表 1 に示す第 4 週までで，Scratch プログラミングについての基礎を習得することができた。この第 4 週までの学習内容は，Scratch の基本的概念であり，その後も Scratch プログラミングを進めていくための基礎としては十分な内容であると考えられる。

第３章　Python とそのプログラミングスキルの習得

第１節　Python について

Python とは、1991 年に登場したインタプリタ型のテキストプログラミング言語である。Scratch のようにあらかじめ用意されたブロックを組み合わせるのではなく、プログラムを文字や記号、数字のみで記述する。Python のインタプリタは、Python 言語でソースコードを入力して読み込み、命令を実行する。作成したソースプログラムをただちに実行できることは、プログラミングの学習者にかかる負荷をコンパイラ型言語と比較して小さくするので、初心者にとってはメリットになる。Python という名前は、イギリスの喜劇集団「モンティ・パイソン」に由来している。

学習者は、Scratch を習得した後に、他のプログラミング言語も体験・習得し、どのプログラミング言語がどのような分野で有効に使えるかを比較・調査してみたいと考えたため、Python の習得を開始した。当該学習者にとって Scratch との大きな違いはテキストでコードを記述する点である。

第２節　学習の進め方

〔概要〕

［プログラミング言語］　Python 3.9

［期間］　2021 年 9 月 15 日～ 2022 年 1 月 11 日

［使用したテキスト］　Al Sweigart 著　相川愛三訳　『退屈なことは Python にやらせよう　ノンプログラマーにもできる自動化処理プログラミング』[6]

［学習方法］

学習者は約 4 か月（15 週）、自習とゼミナール形式の個別指導を受ける方法で Python の学習を行った。18 章で構成されている使用したテキスト [6] には、第 6 章までで Python プログラミングの基本的概念はすべてカバーできており、その後もプログラムを書くための知識としては十分であるとの記

述がある。そこでテキストの章立てにしたがい、第 1 章から第 6 章までを学習する。個別指導は、毎週 1 回の対面指導なので、Scratch を学習したときよりも丁寧なものとなった。

そして、Scratch の学習時には行わなかった学習日誌をつけることにした。学習日誌は週 1 回のペースで指導者に提出する。学習日誌をつけることは日々の学習での振り返りになり、その時点での疑問が記録される。その疑問は学習を進めて行く中、自力で、または指導によって解決することができた。この学習日誌には、学習者の理解とつまずき、そしてその過程での情意的な面も記録されている。指導者は、日誌の記録内容に赤ペンでフィードバックコメントを書き（図 2)、週 1 回のペースで行われるゼミでの個別指導ではソースコードレビューを行なった。ソースコードレビューとは、一般にプログラムの開発者、担当者が書いたソースコードを閲読してセキュリティ脆弱性あるいはそのきざしを読み取る作業である。プログラミング言語習得においても、この作業は効果的なのでテキストのサンプルプログラムや学習者が自分で書いたソースコードを口頭で説明するソースコードレビューを取り入れた。

第３節　Python 習得の過程と成果

学習者はこの Python プログラミングを学習する前に、その習得過程を記録してその成果をまとめて本誌に論文投稿することをあらかじめ決めていた。表 2 には、その学習過程と学習に費やした各回の時間を分単位で示した。これから同じような立場の初学者の学習時間の目安となろう。学習者は自習によって先に学習を進めていくので、しばしば、ゼミでの個別指導では、自習した内容に戻り、詳しく解説することがあった。表中のテキストページの下に「(戻り)」と記載されているのはそういう意味である。また、学習者の感想と疑問、指導者から学習者への応答や指導を簡潔に記している。

Python の学習過程で、学習者にとっての困難やつまずき、それに対しての指導を週ごとに整理して以下に示す。15 週に渡り、自習は 25 時間 15 分、ゼミでの個別指導には 19 時間 15 分、計 44 時間 30 分の学習であった。その成果としては、Python によるプログラミングスキルの、(1) 入出力 (2) フロー制御 (3) 関数 (4) リスト (5) 辞書 (6) 文字列操作 を習得することができた。

図 2 学習日誌へのフィードバック

ブール演算子

and	or	not
かつ	または	否定 ～でない

真偽値　True : ○
False: ×

and演算子，or演算子の真偽値表

A	B	A and B	A or B
○	○	○	○
○	×	×	○
×	○	×	○
×	×	×	×

not 演算子の真偽値表

A	not A
○	×
×	○

図 3　論理演算

●第 1 週目

学習者は、既に Scratch でプログラミングの基本を習得しているので、Python のインストールからコーディング環境づくりまでは自分でできた。まず、Python 入門として、変数への代入と入出力関数の input()、print() を知る。最初のサンプルプログラムをエディタで書いて、ビジュアルプログラミング言語よりもテキストプログラミング言語の方が変数の扱いがわかりやすいという感想をもった。

続いてフロー制御のためには、ブール型（True, False)、比較演算子、ブール演算子を知らなければならない。自習では、テキストのサンプルプログラムを見ながら進めているので順調のようであるが、実際はゼミの指導によって、論理演算、真偽値表についての解説が必要であった（図 3)。この点は、初心者へのプログラミング指導の留意点であろう。フロー制御文は、条件式のブール値によって何を実行するかを決定する。また、ソースコードのブロックを 4 文字スペースの字下げで作る Python の仕様には、慣れるまで戸惑うこともあった。

学習開始から 3 日目において、プログラミングの制御構造の順次と分岐を扱い、if 文、else 文、elif 文 のところで、少しずつ難しくなったと感じた。

●第 2 週目

学習者はプログラミングの制御構造の一つ、反復を while ループ、for ループで記述することを学習する。しかし、終了条件を判断するためのブール値の扱いがよくわからない。そこで改めて、and、or、not の真偽値表とブール値（True, False）の指導を受ける（図 3)。また、While True: は、無限ループを作り、While True !=0: のことで、「!=0」が省略されていることも知る。

この週のテキストの学習内容には、順次処理の重要性を示す例として、if 文と elif 文を続けて記述した 2 種のフローチャートの比較があり、順序の違いによって一方は論理的に到達不可能な例（論理エラーとなっている）を示しているのだが、その意味がよくわからないと日誌に記述した。

●第 3 週目

反復（繰り返し）について、学習者が自力で、for ループと range() 関数を使って、1 から 10 までを表示するコードが書けた日には 2 時間近く集中して自習して、ようやくすっきり理解できた。また、def 文での関数作成を自習し、自分で関数を定義することは便利、見やすいとしながらも複雑で難しいと日誌に記述した。この週の終わりには、ゼミで while ループ文に戻り、break 文、continue 文を復習した。なお、これまでのソースコードレビューで「今までで一番良く説明できた気がする」と記述した。

そして、前週によく理解できなかった if 文と elif 文を続けて記述した例で順序の違いによる論理エラーについて、1 時間くらい考えてやっとわかった。この理解は順次構造を扱うためには不可欠であり、順次処理で真の重要点であるととらえることができた。その詳細は次節（3）を参照されたい。

●第 4 週目

学習者は def 文での関数作成で戻り値と None 値がわからない。また、ローカルスコープとグロー

バルスコープ（また、ローカル変数、グローバル変数）についても、自習で理解するにはかなり難しい。指導者は学習日誌のフィードバックコメントに「関数の外の変数はグローバル変数、関数の中の変数はローカル変数かグローバル変数」と応答した。

また、ソースコードに全角スペースが入っていたためのSyntaxErrorに対し、指導者がゼミで「ごみが入っている」と言ったことに対し、学習者はその意味がわからなかったと日誌に記述している。全角スペースが入っていれば、それが見えないごみとなってSyntaxErrorを起こす。

●第5週目

学習者は「ブラックボックスとしての関数」の意味がわからない。そのフィードバックコメントに「一旦、関数を作ってしまえば、使うときは入力と出力だけを意識すればよく、この関数の中は見えなくてよい（ブラックボックス）」と応答した。

ゼミでは、def文での関数作成の学習にもどって、戻り値とreturn()関数を復習した。また、ローカルスコープとグローバルスコープ（また、ローカル変数、グローバル変数）についての説明を受けたが、その日の学習日誌には「まだはっきりと理解できていない」と記述した。関数の中からグローバル変数を変更したいときのglobal文も十分に理解できなかった。

●第6週目

学習者は関数作成の演習プロジェクトとして、コラッツ数列を出力するプログラムに取り組んだ。入力された整数が偶数なら2で割る、奇数なら3をかけて1をたすcollatz()関数を作成し、関数が1を返すまで呼び出し続けるプログラムである。作成したcollatz()関数は、1回の呼び出しで偶奇判定とそれぞれの処理を行った値を返す。したがってこの関数を繰り返し呼び出すが、関数の処理と繰り返しの処理の関係の理解に苦戦した。

その後、学習者は3日かけてプログラムを完成させることができた。完成後はリストへと進んだ。

●第7週目

学習者は自習でリストの学習を進めたが、学習日誌には「難しくて進められない」と記述した。オブジェクト指向言語でもあるPythonのリストを操作するメソッドや、ミュータブル、イミュータブル、リスト型、タプル型、リストの参照を渡すといったことの理解は自習では難しい。

自習では第4章リストを終えているが、指導者はゼミでリストの最初に戻って、データ構造というのはデータをプログラムで扱うときに格納する形式であり、リストもその一つの形式であることを解説した。その日の学習日誌には「理解がまだ不十分。自習した時よりも構造が少し理解できた。使えるようになったら便利、もう一度この範囲を自分で見直して学びなおしたい」と記述した。

●第8週目

学習者は、絵文字グリッドをかく演習プロジェクトに取り組む。リスト型を使って格子をかく。（イメージとしてはm × n行列である。）学習日誌には「どうコードをかけばよいのかわからない。ループの中にループを使った記述が難しい」と記述した。このとき、指導者は、現在の高等学校の教育課程では行列は扱わないし、学習者は文系なので、なかなかイメージが難しいのではない

かと推測した。

●第 9 週目

学習者は自習で、辞書型のデータ構造に入り、チェスや三目並べをモデル化するのに辞書を用いることを知り、プログラムも面白そうと興味をもった。

ゼミではリストに戻り、自習では難しかった「参照を渡す」について解説した。学習日誌には「参照について、自学よりも理解ができた気がする。（アドレス、住所と考えると分かりやすかった）」と記述した。

●第 10 週目

学習者は自習で、リストの演習プロジェクトで、カンマ付けと文字絵グリッドに取り組む。カンマ付けはリストの値をカンマで区切り、ただし最後は and で区切った文字列で返す。文字絵グリッドは 2 重ループを使って平面の文字絵をかく。学習日誌には「出来たー！」と記述した。

●第 11 週目

自習では辞書型構造で持ち物リストを作る演習に取り組む。for ループで複数代入の技法を使う。また辞書型のメソッドを知る。

●第 12 週目

引き続き持ち物リストの演習に取り組む。

また、自習で文字列操作に入り、文字列操作のメソッドを知る。

●第 13 週目

便利な文字列操作のメソッドを知る。

pyperclip モジュールを用いて文字列のコピー & ペーストをする。

●第 14 週目

自習ではコマンドライン引数のところをやるが、自習での理解は難しい。

ゼミでは、辞書型の演習に取り組む。学習日誌からは、自信もついてきたようすが窺える。

●第 15 週目

ゼミで、第 6 章の文字列操作の解説を行った。文字列を操作するメソッドについて確認した。

表 2　Python の習得過程

	№.	学習日 年月日	テキスト ページ	時間（分）		学習内容	学習者の感想と疑問	※は，指導者による 学習者への応答または指導
				自習	ゼミ			
1週目	1	2021/9/15	pp.3-14	60	60	コーディング環境整備 インタラクティブシェル ファイルエディタ 最初のプログラム	変数の格納や初期化，上書きが Scratchより分かりやすくやりやすい	
	2	2021/9/17	pp.14-31	120		print()等の入出力関数 フロー制御 比較演算子 ブール演算子	コードのブロック， ：（コロン）の意味は?	※ 論理演算 ※ 真偽値表
	3	2021/9/21	pp.31-42 pp.1-12	60	60	フロー制御 elif の順番，論理演算	難しくて少しずつしか 進められなくなってきた	
2週目	4	2021/9/24	pp.13-42 pp.42-51	60 60		コメント付け whileループ，forループ	True, False よくわからない	※ while True:　は， while True !=0: と同じ ※ and, or, notの真偽値表
	5	2021/9/28	pp.23-37		90	フロー制御 if，else，elif	ブール演算子の考え方は表にまとめると考えやすくわかりやすい	※ 図3 ※ 順次，分岐処理 ※ №3で理解不十分の順序の重要性を解説 テキストの vampire.pyとvampire2.pyの違い
3週目	6	2021/9/29	pp.49-55	60		forループ range() モジュールのimport	range()関数で5から0までのカウントダウン range(5,-1,-1)の 第2引数の-1がわからない	※ 終了値より1小さいことから-1になる
	7	2021/10/1	pp.55-57	105		演習問題： forループを使って 1から10までを表示する	プログラムを書く演習問題がしっかり考えられて，最終的に自分の力で書けたので良かった ブロックについて演習問題を解いて確認することで，ようやくスッキリ理解できた	
	8	2021/10/4	pp.59-63	60		def文で関数を定義する 戻り値とreturn文	自分で関数を定義することで便利，見やすい できることがもっと広がりそう でも複雑で難しい	
	9	2021/10/5	pp.40-51 （戻り）		90	whileループ文 break文 continue文 forループとrange()関数	while not name:の訳し方 プログラムの動きなどを今までで一番良く説明できた気がする(ソースコードレビュー)	※ while not name !=0:
4週目	10	2021/10/10	pp.63-71	90		None値 キーワード引数とprint()関数 ローカルスコープ グローバルスコープ， global文	 ローカルスコープとグローバルスコープについて 見分け方，考え方が難しくあまり理解できていない	※ 戻り値 値を返さない戻り値としてNoneを返す ※ 関数の中の変数は， ローカル変数かグローバル変数 ※ 関数の外の変数は， グローバル変数
	11	2021/10/12	pp.52-57 （戻り）		90	forループと同等のwhileループ range()関数の3つの引数 モジュールのインポート from import文 sys.exit()関数	p.55 sys.exit()関数で文法エラーが出たときに、条件式の後ろでDeleteキーを押したこと 先生はごみが入っている?と言っていたがどういうことか	※ 全角スペースが入っていた

	№	学習日 年月日	テキスト ページ	時間（分）		学習内容	学習者の感想と疑問	※は，指導者による 学習者への応答または指導
				自習	ゼミ			
5週目	12	2021/10/18	pp.72-77	90		例外処理 try 〜 except節 数当てゲーム	「ブラックボックスとしての関数」がよくわからない try節は「エラーが起きそうなコードはここですよ」と指示していて，except節は「もし本当にエラーが起きたらこうしてね」を書く	※ 関数の中は見えなくてよい（ブラックボックス）
	13	2021/10/19	pp.59-69 （戻り）		90	パラメータのあるdef文 戻り値とreturn文 None値 キーワード引数とprint()関数 ローカルスコープ グローバルスコープ，global文	ローカルスコープとグローバルスコープについて先生の説明を聞いてなんとなく分かったが，まだはっきりと理解できていない気がする	
6週目	14	2021/10/21	p.78	30		コラッツ数列を表示する（コラッツ関数）	collatz()関数は何となく書けたが，ちゃんと起動しない 最後の1まで表示されない なんでだ	
	15	2021/10/22	p.78	30		コラッツ関数	number = collatz(number) どうしてこれが必要？ 最後の1まで表示させるところが難しかった 偶数と奇数しかないので， 一方（偶数）をifにしたら 他方（奇数）はelseで事足りる	※ while文で繰り返し処理する新しい値をnumberに入れて，collatz()に渡すため
	16	2021/10/24	pp.78-85	60		コラッツ関数でtry 〜 except節 リスト型 インデックス，スライス	try:文を入れる場所を考えるのが難しかった リストについてScratchと似ているところがあった	
	17	2021/10/26	pp.86-92 pp.72-78	30	90	forループとリスト inとnot in演算子 複数代入法 累算代入演算子 メソッド index(), append(),insert() try 〜 except節で例外処理	in， not in演算子と累算代入演算子が面白いと思った なんとなく理解している気でいる コラッツ数列をバグなしで動かすにはあらゆる場合を想定してtry 〜 except文で例外処理を追加しておく必要があること	
7週目	18	2021/10/31	pp.92-103	45		リストのメソッド remove(), sort() ミュータブル イミュータブル タプル型 リストの参照	もっと進めたいけど，難しくて進められない もう少し時間をとって，集中して取り組みたい	
	19	2021/11/2	pp.79-90 （戻り）		90	リスト型 インデックス，長さ 連結 forループとリスト	リストについてまだよくわからず，理解がまだ不十分 自習した時よりも，構造が少し理解できた 特にforループと組み合わせた時が複雑で使えるようになったらめちゃ便利そう 考えにくいけど，もう一度この範囲を自分で見直して学びなおしたい	※ データ構造はデータを扱うときに格納する形式で，リストもその一つの形式，今後「辞書」という形式も出てくる
8週目	20	2021/11/8	pp.101-108	90		リストの参照と複写 m×n行列をリストでかく	やりたいことは分かっているけど，どうコードを書けばよいのかわからない ループの中にループを使った記述が難しい	
	21	2021/11/9	pp.90-96		45	リストのメソッド マジック8ボール ASCIIコード	リストを使うことで，elifコードをたくさん作らずに済む	※ リストのデータ構造のおかげ（リストの有用性）

	№	学習日 年月日	テキスト ページ	時間（分）		学習内容	学習者の感想と疑問	※は，指導者による 学習者への応答または指導
				自習	ゼミ			
9週目	22	2021/11/15	pp.109-124	60		辞書とデータ構造 三目並べ	チェスや三目並べをモデル化するのに辞書を用いること p.124の完全な三目並べのプログラムが面白そうなので，見て自分でも動きを確かめたい	
	23	2021/11/16	pp.96-107 （戻り）		90	ミュータブル，イミュータブル タプル型 参照を渡す copyモジュール copy(), deepcopy() リストの要素をカンマ付け出力 m×n行列の文字絵グリッド	参照について，自学よりも理解が出来た気がする （アドレス，住所と考えると分かりやすかった） 演習プロジェクト難しい〜〜〜 （カンマ付け，m×n行列の文字絵）	
10週目	24	2021/11/21	pp.107-108	60		リストの要素をカンマ付け出力 m×n行列の文字絵グリッド	カンマ付け…繰り返し回数を（len（spam）-1）にして，最後の要素だけ場合分けすること 文字絵グリッド…forループにforループを入れること 分からない　難しい〜〜 でもまだやるまだやれる まだ考えたい	
	25	2021/11/22	pp.124-126	60		辞書とリストの入れ子	分かったようであんまりわかってない気がする	
	26	2021/11/23	pp.107-111 （戻り）		90	リスト型 辞書方 辞書とリストの比較 改行 print('')	出来たー！ 5章の演習プログラムもいっぱい考えて頑張るぞ〜	
11週目	27	2021/11/28	pp.126-128	60		演習： 辞書型構造で持ち物リスト	forループの，変数を2つ設定したときの記述の仕方が今までより分かった気がする	
	28	2021/11/30	pp.127-128 pp.112-117	30	90	持ち物リスト 辞書型のメソッド keys(),values(),items() get(),setdefault() 整形表示	もう少し時間をとって全体的に進めたい 整形表示が便利そう	
12週目	29	2021/12/5	pp.127-130	30		演習： リストから辞書に移す関数 文字列操作 エスケープ文字	演習プロジェクト出来た気がするけどもっといい表記ありそう〜〜 cat'sみたいに入力するときのことを考えてなかった	
	30	2021/12/7	pp.130-140 pp.117-119	60	30	文字列操作のメソッド upper(), lower(), isupper(), islower()他 辞書のデータ構造 三目並べのボード	まだページ残っててへこんだので自習もっと頑張る '12345'.islower()は，英文字じゃないからFalseが出る？	※ ブール値について
13週目	31	2021/12/10	pp.140-143	30		便利な文字列メソッド pyperclipモジュール コピー＆ペーストする	モジュールやらメソッドが多すぎてこれなんだっけ…ってなることが多くなってきた	

	№	学習日 年月日	テキスト ページ	時間（分）		学習内容	学習者の感想と疑問	※は，指導者による 学習者への応答または指導
				自習	ゼミ			
14週目	32	2021/12/17	pp.144-151	75		コマンドライン引数 sys.argv バッチファイル	「もしユーザーが引数を付け忘れたとき（つまり，リストsys.argvの長さが2未満のとき）」って何？ 「バッチファイルを作って，Win-Rキーでファイル名を指定して実行」がよくわからない 全体的にあんまりしっくりきていない 演習プロジェクトも難しそう 困った　やるぞやるぞ	
	33	2021/12/21	pp.124-128 （戻り）		60	辞書とリストの入れ子 辞書のデータ構造の演習 setdefault（）	考える時間が短くなってきた気がする 慣れか理屈が分かってきたのかは分からんけどやったー！	
15週目	34	2022/1/11	pp.129-142		90	文字列操作		

	自習	ゼミ	自習＋ゼミ
合計（分）	1,515	1,155	2,670

自習　25 時間 15 分
ゼミ　19 時間 15 分
合計　44 時間 30 分

第４節　Python の学習で難しかったところとその解決法―学習者による備忘録―

詳しくは Python を学習するためのテキストや Python の仕様書に譲るが、初心者としてのつまずきとその解決法を備忘録として残しておく。初心者のための Tips となろう。

(1) 順次について

プログラミング言語では、プログラムの先頭から順番に命令を実行していく。これを順次処理という。

(2) 分岐について

if 文の節は、条件式が、True（真）のときに実行され、False（偽）ならスキップされる。条件式を評価するために、ブール型、比較演算子、ブール演算子について知らなければならないが、テキストにあるブール演算子と真偽値表の意味がわからず、ゼミで説明を受けた。

ブール型には、True と False の 2 種類のブール値があり、比較演算子、ブール演算子（and, or, not）の評価結果はブール値で返す。まず、比較演算子の扱いには、これまで数学で等号や不等号を使っているので困らなかったが、ブール演算子については、高等学校の数学では、集合演算で「かつ」「または」と否定を扱っているものの、論理演算として、and, or, not のそれぞれについての真偽値表の理解が必要であった（図 3）。

(3) 順次処理の本質としての重要点

テキスト [6] の p.36, p.38 には、4 つの条件式 ①名前が Alice か、②年齢が 12 歳未満か、③年齢が 2000 歳より大きいか、④年齢が 100 歳より大きいか　で、Alice、お嬢ちゃん、バンパイア（不死身

の吸血鬼)、お婆ちゃんのいずれかに振り分けられるフローチャートが例示されている。if 文、elif 文を使って、条件分岐を（A）①②③④ の順に設定した例と（B）①②④③ の順に設定した例の 2 つがあり、同じ 4 つの条件なのに、（A）から（B）のように順序を変えるとバグを作ることになるという意味がわからなかった。むしろ、条件の年齢を（B）のように小さい順に並べる方がきれいだとさえ思った。

1 週間程かけて考え直しをしたとき、Alice でない 3000 歳を想定することによって、（A）の順と（B）の順の違いが明白になった。3000 歳なら不死身の吸血鬼へと振り分けたいが、（B）の順では、①は Alice でないので False、②では 3000 は 12 より大きいので False、④では 3000 は 100 より大きいので True となり、3000 歳の吸血鬼を炙り出すことができずに、お婆ちゃんへと振り分けてしまう。

順次とは順番に命令を実行していくことであるが、順序への注意を意識しなければならないこのような例こそが、順次処理の本質として重要であることがわかった。

(4) 反復（繰り返し）について

一定の回数だけコードのブロックを実行したいときに for ループと range() 関数を用いて、for i in range(5): のように書く。すると、for ループの節のコードは 5 回繰り返されるが、最初に、変数 i に 0 が設定される。また、フロー制御文（この例では、for 文）はコロン（:）で終わり、次の行からコードのブロックを字下げして（for 節を）書く。つまり、これらを簡潔に整理すると、

① 0 ～ 4 までで、5 回繰り返されること
② フロー制御文はコロン（ : ）で終わる
③ ブロックの区間は、コードの字下げで指定する

のようになるが、この 3 つを正しく理解する必要があった。

(5) ローカルとグローバルについて

始めの頃は、ローカルスコープとグローバルスコープの考え方や、分ける意味や良さを理解することができなかった。徐々に理解を重ねて行くことで、次の (I) ～ (IV) に集約されることがわかった (テキスト[6]の pp.65-71)。しかし、まだ難しいと感じる。迷ったときには、再びここに立ち戻る必要がある。

（Ⅰ）グローバルスコープについては、
・一つだけ存在し、プログラムが実行されるときに生成される
・プログラムが終了すると消滅する
・グローバルスコープの中の変数をグローバル変数という

（Ⅱ）ローカルスコープについては、
・関数が呼び出されるたびに生成される
・関数から返ると消滅する
・ローカルスコープの中の変数をローカル変数という

（Ⅲ）スコープの扱いについては、

・グローバルスコープのコードは、ローカル変数を使うことができない
・ローカルスコープからはグローバル変数にアクセスできる
・関数のローカルスコープのコードは、他の関数のローカルスコープの変数を使うことができない
・スコープが異なれば、異なる変数に同じ名前を使ってよい

（Ⅳ）global 文

・関数の中で、変数はグローバル変数かローカル変数のどちらかであり、関数の中でグローバル変数の値を変更したいときには、必ず global 文を用いる必要がある

(6) データ構造について

データ構造であるリスト、タプル、辞書が次々と出てきた当初は困惑した。今は、すべて複数の値を格納でき、大量のデータを扱うのに役立つデータ構造であると理解している。

また、リストとほぼ同じデータ型のタプルについて、両者の違いが何のための違いなのかに困惑し、とくに、それぞれが、ミュータブル（変更可能）とイミュータブル（変更不可）であるといっても、その実用のための理解までには至らなかった。リストに対しタプルは、要素の追加、削除、置換ができないことにその本質があるのだろう。データ構造の違いを目的や状況に応じて適切に使用するのが難しい。

(7) 予約語やキーワード と英単語

Python は当然のことながら英語がベースとなっている。組み込み関数やメソッドの機能は、その名称と元の英単語と照らしながら把握することも重要である。このことは、自分で変数名や関数名をつけるときにも同様で、意味をもった命名によって、後でソースコードを見るときにもわかりやすい。

(8) 理解につながるこつ

Python の学習は難しく、正しく理解することに時間を要することもあったが、正しい理解につながるこつが 3 つあった。

① ノートなどに書き出して考えを整理する

テキスト [6] にあった演習プロジェクトなどで、ループや関数などを複数記述する必要がある場合に、漠然とコードを打ち込むのではなく、まずやりたいことは何か、その優先順位、それをするために何が必要かを考えて要点を整理する。このように考え方や思ったことをメモして、書きたいコードを言葉で整理していくことは、大いに役立った。

例えば、図 4 は、持ち物のリストにそれぞれの所有数を合わせて辞書のデータ構造で管理する関数を作成する演習に取り組んだときのメモである。そこには、1. 定義する関数の仕様を書き出し、2. for 節での処理、3. 使用するメソッドとその特徴的な機能、さらに、4. 累算代入演算子を使っ

た短縮形を用いることを記している。

② Python 用エディタの「IDLE」のインタラクティブシェルにコードを入れてみて、実際に試してみる

Python が理解できないコードや指示が含まれていると、エラーメッセージを表示して動作が停止する。エラーメッセージは、どこがエラーの内容なのか伝えてくれる。そのため、間違えてもエラーメッセージを参照すれば、正しいコードの記述につながる。インタラクティブシェルに入力してみて、部分的に動作確認をしながら、間違っても気にせず、とにかく試してみることが理解と定着につながった。

③ ソースコードレビューを行う

ソースコードを口頭で説明することで、そのコードで正しく動作するかを確認した。コードの記述とその動作を真に理解していなければ、他者に理解してもらうことは難しいので、ソースコードレビューは自分自身の書いたコードの働きと動作を確かに理解するために役立った。

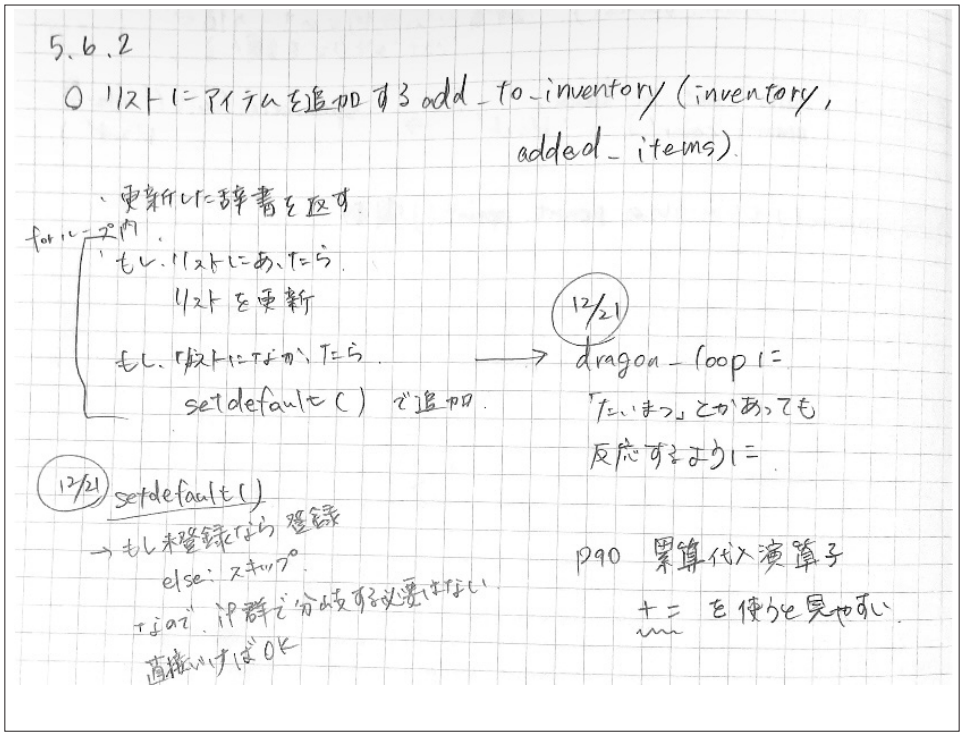

図 4　コーディングのための学習者のメモ

第４章　「ことばの力の育成」におけるプログラミングの有用性

第１節　勤務校の実態、児童の様子

本稿筆頭著者が現在勤務している小学校（以下、勤務校）の児童たちは、非常に素直で自ら積極的

に人に話しかけたり、交流したりしようとする人懐っこさがある。その一方で、自由に表現できる語彙が少なく、感想を発表する際には「すごい」という表現ばかりで、具体的に「どのように」すごいのか、「すごい」ということを自分なりの感じ方によって「素敵」「感動した」などの表現に言い換えて発表することができない。また、児童らが使用する語彙にはプラス表現の言葉よりもマイナス表現の言葉の方が多い。嫌なことがあると、「何が嫌か」「どのように嫌か」を伝える前に「うざい」「死ね」という言葉が出てしまい、児童間で誤解が生じてアクシデントになってしまうことが多々ある。また学習面では、多数の情報が含まれた問題文を読み、必要な情報を自ら取得・精査して問題に答えることが苦手で難しく、そこには知識不足だけでなく、読解力が弱いという課題がある。

第２節　「ことばの力の育成」について

勤務校では、児童が自分の思いを自分の言葉で伝えることができるように「ことばの力の育成」を教員研修課題としている。そのために、低学年、中学年、高学年のそれぞれの段階で目指すべき態度を設定している。それらは以下のとおりである。

【低学年】　・プラス表現の言葉を多くする
　　・その言葉や表現を教員が認める
【中学年】　・人の意見を聞いて受け止める
【高学年】　・意見を受け止めて考えをさらに高め合う

「ことばの力の育成」では、語彙や国語的な表現だけでなく、順を追って説明する力や、自分の考えをかみ砕いて効率よく表現する力を重視している。特にその力を育てるために、学校内で統一した発表の仕方のモデルを示し、要点を整理したり、物事の順序を考えて伝えたりすることに取り組んでいる（図 5)。この「発表のモデル」にしたがうと、自分の考えとその根拠、考えを導いた思考の道筋を明確にすることができ、児童がその思考を言語化するときの助けになると考え、学校全体で取り組んでいる。

私は、（自分の考え）　だと思います。
なぜなら、（そう考える理由　）　だからです。

図 5　発表のモデル

第３節　「発表のモデル」とプログラミング

この「発表のモデル」は、(A) 自分の考え と (B) そう考える理由 から成る。そこで、このモデルとプログラミング構造との関係づけを行ってみたい。

(A) が洗練されたものであるためには、①要点を絞る、②順序づける、③優先順位をつける、④無駄をなくす、⑤簡潔にする　などを行わなければならないであろう。①要点を絞って、②順序づけることは、プログラミングの基本構造の順次であり、さらにいくつかある項目に③優先順位をつけると

いうことは、条件によって分岐することにつながる。（B）は、（A）のための根拠となるので、根拠を明らかにした考えというのは、論理的思考の一つの形式である。また、論理的思考では、「そして」「または」「～でない」「もし～ならば」などのことばを使い、「そして」「または」「～でない」は、それぞれ、and、or、not の論理演算であり、「もし～ならば」は、if 文による分岐である。また、④無駄をなくす、⑤簡潔にする、ためには反復（繰り返し）を用いることもありえる。このように見ると、この「発表のモデル」は、プログラミングの順次、分岐、反復の３つの制御構造に照らすこともできる。そうすると、「発表のモデル」にしたがって、「ことばの力の育成」を目指す勤務校の取り組みと同様に、プログラミングによっても「ことばの力の育成」を目指すための何らかの寄与が期待できるのではないだろうか。

実際、自身がプログラミング言語の学習を経験する中で、論理的に考えたり、整理したりする力や、ことばで表現する力を向上させることができたと実感している。そして、それらはプログラミングをしているときだけではなく、日常生活においても働き、そこでの要件とその優先順位を考えるようにもなった。

第５章　まとめと考察

本研究では、「小学校プログラミング教育を円滑に実施するために教師にプログラミングスキルが必要である」と仮定し、一般的な私立女子大学で小学校教員を目指す学生だった本稿筆頭著者が、まず自ら被験者となり、プログラミング言語習得にチャレンジした。学部在籍時にビジュアルプログラミング言語 Scratch を習得し、その後、約１年経って専攻科在籍時にテキストプログラミング言語 Python を習得した。Python を学習したことで、Scratch を習得したときよりも、さらにプログラミングについての理解度は高まった。Python では、具体的に、入出力、フロー制御、関数、データ構造（リスト、辞書）、文字列操作までを習得できて、プログラミングについての基本的概念はカバーできた。その後も続けてプログラムを書くために必要なことがあれば、自ら調べながら十分に対応できるレベルである。

『小学校プログラミング教育の手引』には、小学校プログラミング教育のねらいは、①「プログラミング的思考」を育むこと、②プログラムの働きや良さ、情報社会がコンピュータ等の情報技術によって支えられていることなどに気付くことができるようにするとともに、コンピュータ等を上手に活用して身近な問題を解決したり、よりよい社会を築いたりしようとする態度を育むこと、③各教科等の内容を指導する中で実施する場合には、各教科等での学びをより確実なものとすることとされている。

①の「プログラミング的思考」を自分が意図する一連の活動を実現するために３つの制御構造である順次、分岐、反復だけで構成していく力とするなら、指導者となる教員には、それら３つの語の意味を上辺だけでとらえるのではなく、実際にそれらの本質を理解する必要があると考える。そのためには、第３章第４節に示した内容までを必要な範囲とし、その習得過程で起こるようなつまずきを克服してこそ、正しくその本質まで理解できるようになると考える。さらに、②のプログラムの働きや

良さ等に気付き、コンピュータ等を上手に活用する態度を育み、③の各教科等での学びをより確実なものとするためにプログラミングを実施するには、指導者となる教員にもプログラミングスキルが必要であると考えるのは自然ではないだろうか。そして自らの経験からは、小学校教員としてその必要性を自覚し、そのスキルを習得する意思があるなら、自習でも習得は可能であると考える。そして、もし適切な指導を受けられるなら、それも望ましい。

もう一つの「プログラミングを学ぶことでも日常の言語能力の育成に寄与するのではないか」という仮説についても肯定的に支持したい。日常の言語を使って、自分の考えを筋道立てて他者に伝えるためには、要点を絞る、順序づける、優先順位をつける、無駄をなくす、簡潔にする　など行いそれらを整理して表現しているが、プログラミングをするときも、同様の思考活動を行っていたと実感しているからである。そして、プログラミングを習得したことによって、逆に、日常生活においても、プログラミングをしているときのように、そこでの要件とその優先順位を考えるようになったからである。したがって、国語や生徒指導的な面からだけではなく、プログラミングの指導を通して、自分なりに子供たちの「ことばの力の育成」にもアプローチしていきたい。

引用・参考文献

[1] 文部科学省（2016）「幼稚園、小学校、中学校、高等学校及び特別支援学校の学習指導要領等の改善及び必要な方策等について（答申）（中教審第 197 号）」,
https://www.mext.go.jp/b_menu/shingi/chukyo/chukyo0/toushin/1380731.htm,（最終閲覧日：2022 年 10 月 09 日）.

[2] 文部科学省（2016）「小学校段階におけるプログラミング教育の在り方について（議論の取りまとめ）」,
https://www.mext.go.jp/b_menu/shingi/chousa/shotou/122/attach/1372525.htm,（最終閲覧日：2022 年 10 月 09 日）.

[3] 中植正剛, 太田和志, 鴨谷真知子（2019）『Scratch で学ぶプログラミングとアルゴリズムの基本』, 日経 BP.

[4] 文部科学省（2018）『小学校プログラミング教育の手引（第二版）』

[5] 坪谷諭泉那（2021）「算数の学びを支援するためのプログラミングの効用とプログラミングを用いた授業づくり」, 神戸女子大学文学部教育学科卒業論文.

[6] Al Sweigart 著・相川 愛三訳（2017）『退屈なことは Python にやらせよう―ノンプログラマーにもできる自動化処理プログラミング―』, オライリー・ジャパン.